EEEM007 ADVANCED SIGNAL PROCESSING
PATTERN RECOGNTION

Yi Zhou 6567008

EXPERIMENT 1

1.1 Descriptions

In this experiment, the effect of traning sample size on the classifier performance is studied. Several Gaussian classifiers are designed using different number of training samples. Their performances on traning set and test set are recorded and compared. Moreover, the choice of k for k Nearest Neighbours (k-NN) classifier with respect with the training sample size is studied. Several k-NN classifiers are trained as size of traning set varies. The optimal k which gives the smallest test error is recorded and analysed.

1.2 Theoretical Analysis

The Gaussian classifier is trained using the training set. The mean and the variance in each class of data is estimated, then the Gaussian model is built with respect to the mean and the variance. Since we generate the same number of data for each class, the prior probability is 0.5 for both classes, thus we can neglect the prior. Then, we assign the class ω_i to the test sample x according to the decision rule

$$x \to \omega_i \quad if \quad P(\omega_i|x) \; = \; max_{j=1}^{m} P(\omega_j|x)$$

The performance of the classifier relies on the quality of training set. If training set is small, the classifier is likely to overfit the local structure. Then the classifier could have small error in traning set but cannot generalize well in test set. As we feed more data into the classifier, the generalization will improve.

The k-NN classifier make the classification of a test sample according to the categories of the first k closest data in the training set.

$$x \to \omega_j \quad if \quad k_j = max_{i=1}^{m} k_i$$

where k_i denotes the number of samples from class i among the k-nearest neighbours to x. The distance metric is usually selected as the Euclidean distance. One key problem for k-NN is to determine the hyper-parameter k. If the value of k is too small, the tend to overfit the local structure and noisy data. If the value of k is too large, the classifier could be less discriminative, because it could include the data from other classes into account. In practice, the value of k should be selected as an odd number to ensure there is always a major class. As the size of training set increases, the value of k should be larger to capture the global structure of the dataset. However, it cannot exceed the overall number of data in the training set.

1.3 Experimental Results

In this experiment, two classes of data, each contains 100 samples, are sampled from two Gaussian distributions. Class 1 has mean $\mu_1 = [3,2]$ and covariance $\Sigma = [4,2; 2,4]$. Class 2 has mean $\mu_2 = [6,5]$ and the same covariance Σ. The data distribution is shown in figure 1.1. The According to the assignment, the Bayes minimum error is 0.1922.

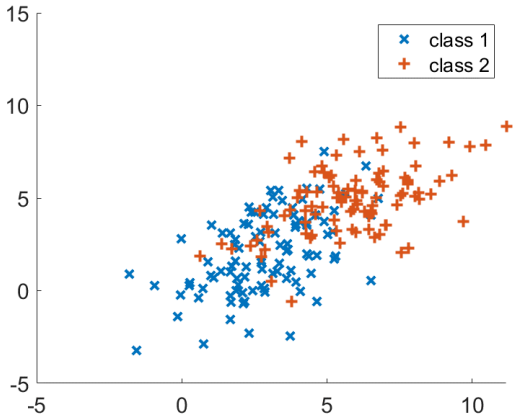


Figure 1.1

In the first experiment, several Gaussian classifiers are designed using different number of training samples. The sample size is chosen from [3,5,10,50,100]. The corresponding training error probability is recorded. Next, the classifiers are tested in the full test set of 100 samples and the test probabilities are also recorded. Figure 1.2 shows the effect of training sample size on the classifier performance. If we train the Gaussian classifier with a small amount of data, the classifier tends to overfit the training data locally, resulting a smaller error probability than Bayes error. However, if we test the classifier in the test set, we find the test error probability is high, indicating the classifier cannot generalize well on the whole dataset. If we increase the number of training set, we can find the training error decreases and the test error increase, and finally these two errors reach plateau. Since there are not enough data, these two errors do not converge to the Bayes error. The overfitting phenomenon still exists but less severe.

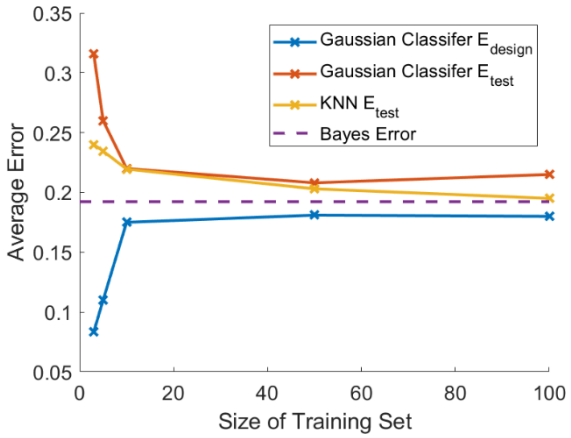
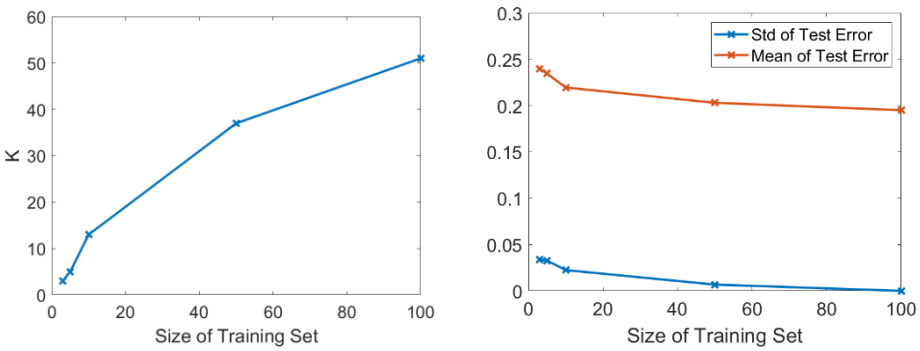


Figure 1.2

In the second experiment, several k-NN classifiers are trained as size of traning set varies from 3 to 100. The optimal k which gives the smallest test error is recorded. Figure 1.3 (a) shows the value of k which gives the smallest test error as the size of training set increases. The corresponding mean and standard deviation of test error probabilities are shown in figure 1.3 (b). When the training set is small, let k equal the training data per class gives smaller error

probability. The test error probability has a high mean and a high variance, indicating the classifier is strongly affected by the representative of the training data. From figure 1.2, compared to the test error of Gaussian classifier, the k-NN classifier has smaller test error probability. The reason is an outlier will only affect k-NN locally but has a big effect on the shape of Gaussian distribution. As the size of traning set increases, the optimal k also increases but not linearly. This is reasonable since the far-distance data, for example the data in the low-density area of a Gaussian distribution, contribute noise in the classification, thus can be considered as outliers and neglected in the classification. From figure 1.2, we can find the k-NN classifier has a lower test error probability than Gaussian classifier when training set is large. This again prove a smaller k make the k-NN classifier insensitive to outliers in the training set, while Gaussian classifier needs to consider them all. From figure 1.3 (b), as the training size increases, the mean decreases to close to Bayes error and the variance converges to zero, indicating the classifier is stable and has a good performance.



(a) (b)

Figure 1.3

1.4 Conclusions

Generally, a small training set is more likely to be unrepresentative, which makes the trained classifier biased and unstable. As we use more training samples, the generalization of the classifier improves. For Gaussian classifier, if the training size is small, it tends to overfit the training set and thus has a poor performance on test set. It is more sensitive to the representative of training data than k-NN classifier. For k-NN classifier, if training set is small, the optimal k tends to be chosen as the number of data per class. However, if the training set is large enough, neglects some unrepresentative data, i.e. choose a smaller k, can give a more robust classifier, and the accuracy is also higher.

EXPERIMENT 2

2.1 Descriptions

In this experiment, the effect of the size of test set on the reliability of the empirical error count estimator are studied. A well-trained classifier is tested in the test set with different sizes. The mean and the variance of the test error probability are recorded. Moreover, suppose we have a fixed number of data, the training-test data split strategy is studied through trying different split ratios and recording the training/test error probabilities.

2.2 Theoretical Analysis

The test set will not participate in the traning process. Thus, it only affects the evaluation of the trained classifier. Ideally, we want the classifier can predict every test example correctly. However, since the class distributions always overlap, the prediction cannot be 100% accuracy. The optimal error, i.e. the lower bound of error, is given by the Bayes error. In this example, the Bayes error is 0.1922. Therefore, the test set should reveal the data distribution of each class. Since we sample the test data uniformly from the dataset, the more data in the test set, the more representative the test set can be. If the size of the test set is too small, then the variance of the test errors could be large. The mean of the errors is likely to be smaller than the Bayes error, if we sample the test data uniformly. The reason there are more samples far away the decision boundary, i.e. the simple cases, than the hard cases around the decision boundary. As we use more and more data in the test set, the variance should drop and the mean should converge to the value near Bayes error.

If we have a limited amount of data, the train-test split should consider the trade-off between training an unbiased classifier and making an unbiased evaluation. If we have a small ratio of training data, the classifier is very likely to overfit the local data structure. If we have a small ratio of test data, as mentioned before, the evaluation could be biased. Intuitively, the training process is more important in a practical task. Thus, the split rule is, on the premise of enough training data for an unbiased classifier, we want to find the most suitable amount of test set split for evaluation.

2.3 Experimental Results

In this experiment, two classes of data, each contains 100 samples, are sampled from two Gaussian distributions. Class 1 has mean $\mu_1 = [3,2]$ and covariance $\Sigma = [4,2; 2,4]$. Class 2 has mean $\mu_2 = [6,5]$ and the same covariance Σ. A gaussian classifier is trained on these data. Then, I generate 10 independent test sets, each contains 100 samples per class and is generated from the same Gaussian distributions as the training set. Next, I evaluate the classifier based on the test samples with different size within [3,5,10,50,100]. The evaluation is processed 10 times in each data sets, and the mean and the variance of test errors are recorded. Figure 2 shows the experimental results. It should be noticed that the mean of test error is affected by the random seed. Thus, the shape of the curve could vary as a different random seed. However, the trend is the same, that is firstly an inaccurate mean (either too large or too small) with high variance and then converge to the Bayes error with small variance.

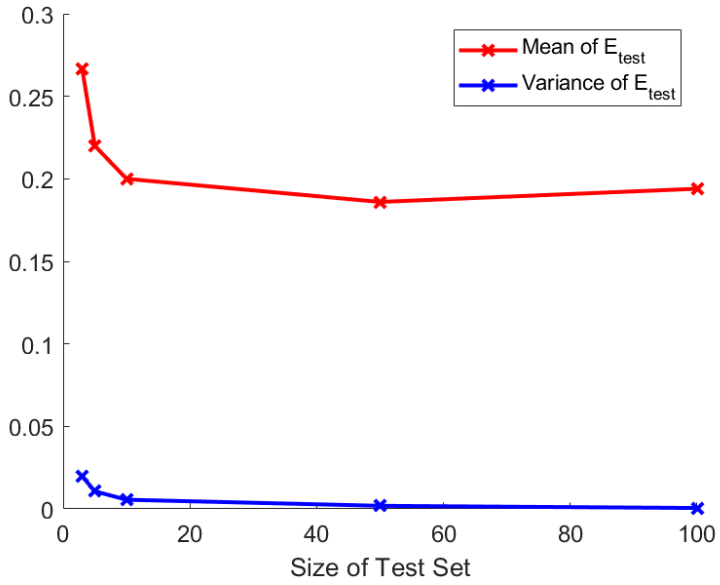


Figure 2

In the second part of the experiment, there are total 100 available data per class. Then we split the data into training set and test set following the ratio in the table below. We train a Gaussian classifier in the training set and evaluate it using the corresponding test set. From the table, we can find that as the ratio of training data increases, the variance in design error decreases while the variance in test error increases. Specifically, the design error has a biased mean and high variance if the size of training set is smaller than 50. The bias can be caused by either overfitting to local structure or underfitting due to unrepresentative data. The last three options: 50:50, 75:25 and 90:10 are all available choices. Both the design error and the test error are close to the Bayes error. Considering the trade-off, the best choice could be 75:25.

Size of Training Set	Size of Test Set	Mean Design Error	Mean Test Error	Std Design Error	Std Test Error
3	97	0.1500	0.2031	0.1657	0.0055
5	95	0.2100	0.2053	0.1197	0.0043
10	90	0.2450	0.2083	0.0438	0.0088
50	50	0.1920	0.2200	0.0355	0.0394
75	25	0.1853	0.1940	0.0140	0.0597
90	10	0.1844	0.2000	0.0125	0.0913

2.4 Conclusions

If we use too few data in test set, the test error probability has a high variance and cannot reveal the real performance of the classifier. As we use more and more data in the test set, the variance should drop and the mean should converge to the true value. If we have a limited number of data, we should consider the trade-off between using enough training data to avoid overfitting and using enough test data to properly evaluate the performance. Since training is more important and more sensitive to data, we tend to adjust the ratio of test data under the premise of enough training data.

EXPERIMENT 3

3.1 Descriptions

In this experiment, the dependence of test error on the dimensionality of the pattern recognition problem is studied. Firstly, three Gaussian distributions with dimension of 5, 10 and 15 are built. Then, we duplicate these distributions and move the new ones to form the overlaps which corresponds to a specific error probability. Next, we train the Gaussian classifiers for each case with different size of training set. The test error probabilities of these classifiers are compared.

3.2 Theoretical Analysis

In high dimensional space, most of the volume of a sphere is in a narrow annulus [1]. Assume the Gaussian distribution takes the form of

$$p(x) = \frac{1}{(2\pi)^{d/2}\sigma^d} exp\left(-\frac{|x|^2}{2\sigma^2}\right)$$

Where x is a d-dimensional vector. Although the density function has its maximum at origin, the volume in the high-dimensional sphere is almost zero. According to [1], the probability mass is concentrated on the surface whose radius is $\sqrt{d}\sigma$. If the radius is less than $\sqrt{d}\sigma$, the density is high but the volume is nearly zero, thus the integration yields to zero. If the radius is over $\sqrt{d}\sigma$, although the volume increases, the density is very low, thus the integration will not increase. Therefore, only a small part of samples in the high dimensional space contributes to the estimation of the Gaussian density. In other words, the estimation needs extremely large number of samples in high dimensional space.

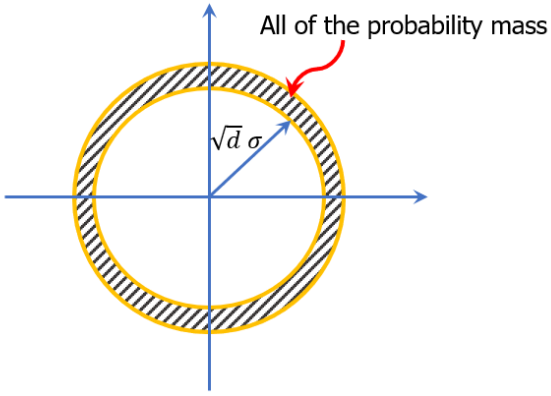


Figure 3.1

3.3 Experimental Results

Firstly, I generate three Gaussian distribution with 5, 10 and 15 dimensions. The means are all located at origin and the variances are identity. Then, I move the distributions to generate the second class of data. Since determination of the true error probability in high dimensional spaces is hard, I take the estimated error of 500 samples as the true error. Next, I literally adjust the distance between two distributions in each case to ensure the estimated true error is 0.06. During the process, I find I need to move a shorter distance in the high-dimensional space (distance of 1.25 for 15 dimensions) compared to the necessary distance in the lower-dimensional space (distance 1.89 for 5 dimensions). In other words, the distributions have smaller overlaps and are easier to separate in high-dimensional space.

Then, in each case, some Gaussian classifiers are trained with the training set whose size is chosen from [3,5,10,20,50,100,200]. Finally, the performances of these classifiers are evaluated in the test set with 500 samples. The results are shown in figure 3. We can find the curves are steeper and fast converge in low dimensional space. In higher dimensional space, it needs more training data to achieve the same test error probability.

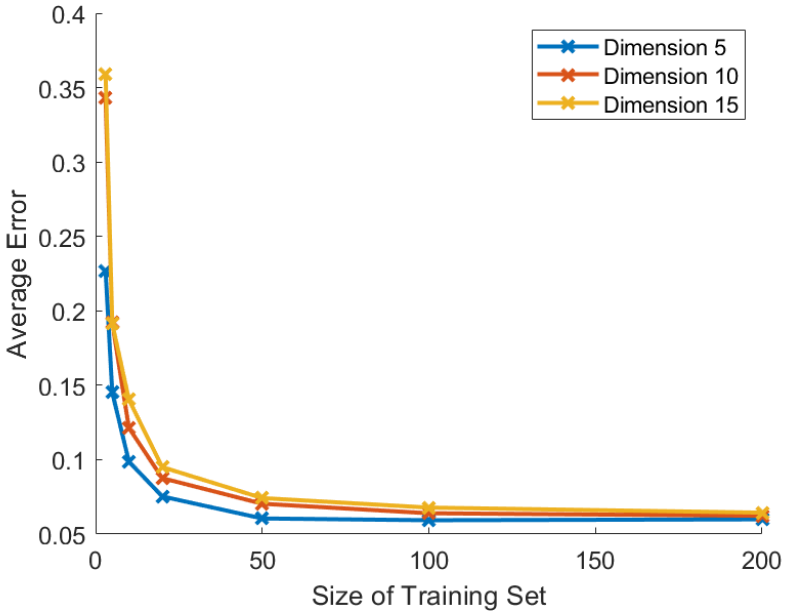


Figure 3.2

I record the required size of training set when the error probability can be rounded into 0.6 in each dimensional space. As we can see from the table, as the dimension increases, we need more training data to achieve the same accuracy. Although we have only a few examples, we can infer that the required size grows exponentially as the dimension increases.

Dimension	Required Size of Training Set
5	40
10	100
15	200

3.4 Conclusions

If the inter-centre distance is fixed, the distributions in the high dimensional space have smaller overlaps and are easier to separate. If the true error probability of separation is fixed, to achieve the same classification error probability, the classifier needs more training data in high dimensional space than that in the lower dimensional space. The required size grows exponentially as the dimension increases.

Reference

[1] CS4850 Mathematical Foundations for the Information Age
https://www.cs.cornell.edu/courses/cs4850/2020sp/

EXPERIMENT 4

4.1 Descriptions

In this experiment, the relationship between class separability and error probability is studied. The Mahalanobis distance is used to measure the class separability of two normal distributions with the same covariance. As move one distribution away from the other, the Mahalanobis distance and the training/test error of the Gaussian classifiers are recorded and analysed.

4.2 Theoretical Analysis

The probabilistic distance between two normal distribution is given by

$$d = \frac{1}{2}(\mu_2 - \mu_1)^T(\Sigma_1^{-1} + \Sigma_2^{-1})(\mu_2 - \mu_1) + Tr\{\Sigma_1^{-1}\Sigma_2 + \Sigma_2^{-1}\Sigma_1 - 2I\}$$

If $\Sigma_1 = \Sigma_2 = \Sigma$, then the probabilistic distance degenerates into the Mahalanobis distance, which is given by

$$d = (\mu_2 - \mu_1)^T\Sigma^{-1}(\mu_2 - \mu_1)$$

If two distributions are getting away from each other, the Mahalanobis distance between them gets larger. Therefore, this distance can be used to measure the class separability of two normal distributions. According the Bayesian decision rule, if two distributions are far away each other, the overlapped area is small, thus the Bayes optimal error is also small. Therefore, we can infer the error probability is negative related to the class separability.

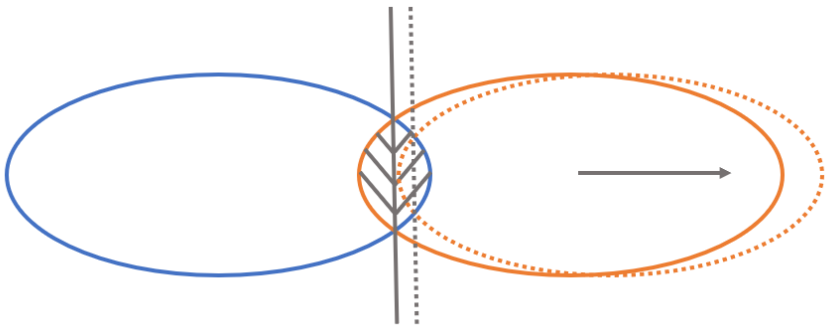


Figure 4.1

4.3 Experimental Results

In this experiment, I generate two classes of data from two 2-dimensional Gaussian distributions. Class 1 has mean $\mu_1 = [3,2]$ and covariance $\Sigma = [4,2;2,4]$. Class 2 has mean $\mu_2 = [6,5]$ and the same covariance Σ. Since the covariance is same, the distance between these two classes can be measured by the Mahalanobis distance. Then, I move the second distribution away the first distribution along the vector $[1,1]$ with step size of $\sqrt{2}$. At each step, the Gaussian classifier is applied to fit the training data and make the classification in the test set. The figure below shows the relationship with classification error and the Mahalanobis distance. As the distance increases, the classification error in both training set and test set decrease until becoming zeros. The experimental result proves the above theoretical analysis. As the distance

increases, the class separability increases, the overlap between two distributions decreases, thus the error probability decreases.

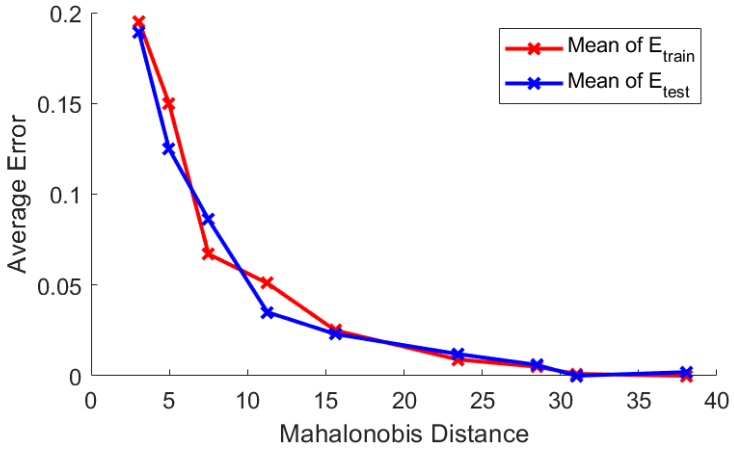


Figure 4.2

4.4 Conclusions

Mahalanobis distance can be used to measure the class separability between two normal distribution with the same covariance matrix. As the distance increases, the classification error in both training set and test set decrease until becoming zeros.

APPENDIX

Q1 Codes

```
mu1 = [3, 2];
mu2 = [6, 5];
cov1 = [4, 2; 2, 4];
cov2 = [4, 2; 2, 4];
N = 100;
N_D = [3, 5, 10, 50, 100];
loop = 10;
rng(3)  % For reproducibility
design_data = [mvnrnd(mu1,cov1,N); mvnrnd(mu2,cov2,N)];
design_label = [ones(N,1); zeros(N,1)];
test_data = [mvnrnd(mu1,cov1,N); mvnrnd(mu2,cov2,N)];
test_label = [ones(N,1); zeros(N,1)];
figure(1)
hold on
plot(design_data(1:N,1),design_data(1:N,2),'x','Linewidth',2,'Markersize',8)
plot(design_data(N+1:2*N,1),design_data(N+1:2*N,2),'+','Linewidth',2,'Markersize',8)
legend("class 1","class 2")
err_design = zeros(length(N_D), loop);
err_test = zeros(length(N_D), loop);
for i = 1:length(N_D)
   for j = 1:loop
      ind = [randsample(N, N_D(i)); randsample(N, N_D(i))+N];
      train_data = design_data(ind,:);
      train_label = design_label(ind);
      g_classifer = fitcnb(train_data,train_label);
      err_design(i,j) = mean(abs(train_label - round(predict(g_classifer, train_data))));
      err_test(i,j) = mean(abs(test_label - round(predict(g_classifer, test_data))));
%       if sum(train_label) < 2
%          err_design(i, j) = 0;
%          err_test(i, j) = mean(abs(test_label - 0));
%       elseif sum(train_label) > 2*N_D(i) - 2
%          err_design(i, j) = 0;
%          err_test(i, j) = mean(abs(test_label - 1));
%       else
%          g_classifer = fitcnb(train_data,train_label);
%          err_design(i,j) = mean(abs(train_label - round(predict(g_classifer, train_data))));
%          err_test(i,j) = mean(abs(test_label - round(predict(g_classifer, test_data))));
%       end
   end
end
err_design_mean = mean(err_design,2);
err_test_mean = mean(err_test,2);
figure(2)
hold on
plot(N_D, err_design_mean, 'rx-','Linewidth',2,'Markersize',8)
```

```
plot(N_D, err_test_mean, 'bx-','Linewidth',2,'Markersize',8)


err_knn_min = 2*N*ones(length(N_D),1);
k_min = zeros(length(N_D),1);
for i = 1:length(N_D)
   for k = 1:2:min(51, 2*N_D(i) -1)
      err_knn = zeros(loop,1);
      for j = 1: loop
         ind = [randsample(N, N_D(i)); randsample(N, N_D(i))+N];
         train_data = design_data(ind,:);
         train_label = design_label(ind);
         knn_classifier = fitcknn(train_data,train_label,'NumNeighbors',k);
         err_knn(j) = mean(abs(test_label - predict(knn_classifier, test_data)));
      end
      if mean(err_knn) < err_knn_min(i)
         err_knn_min(i) = mean(err_knn);
         k_min(i) = k;
      end
   end
end
figure(2)
hold on
plot(N_D, err_knn_min, 'gx-','Linewidth',2,'Markersize',8)
xlabel("Size of Training Set")
ylabel("Average Error")
legend("Gaussian Classifer E_{design}","Gaussian Classifer E_{test}", "KNN E_{test}")
figure
plot(N_D, k_min, 'x-','Linewidth',2,'Markersize',8)
xlabel("Size of Training Set")
ylabel("K")
```

Q2 Codes

```
mu1 = [3, 2];
mu2 = [6, 5];
cov1 = [4, 2; 2, 4];
cov2 = [4, 2; 2, 4];
N = 100;
N_T = [3, 5, 10, 50, 100];
loop = 10;
rng(37)  % For reproducibility
design_data = [mvnrnd(mu1,cov1,N); mvnrnd(mu2,cov2,N)];
design_label = [ones(N,1); zeros(N,1)];

ind = [randsample(N, 100); randsample(N, 100)+N];
train_data = design_data(ind,:);
train_label = design_label(ind);
g_classifer = fitcnb(train_data,train_label);
```

```
err_design = mean(abs(train_label - round(predict(g_classifer, train_data))));
err_test = zeros(length(N_T),loop);

for l =1:loop
   test_data= [mvnrnd(mu1,cov1,N); mvnrnd(mu2,cov2,N)];
   test_label = [ones(N,1); zeros(N,1)];
   for i = 1:length(N_T)
      ind = [randsample(N, N_T(i)); randsample(N, N_T(i))+N];
      test_data_subset = test_data(ind,:);
      test_label_subset = test_label(ind);
      err_test(i,l) = mean(abs(test_label_subset - round(predict(g_classifer,
test_data_subset))));
   end
end
err_test_mean = mean(err_test,2);
err_test_var = var(err_test,0,2);
figure(1)
hold on
plot(N_D, err_test_mean, 'rx-','Linewidth',2,'Markersize',8)
plot(N_D, err_test_var, 'bx-','Linewidth',2,'Markersize',8)
xlabel("Size of Test Set")
legend("Mean of E_{test}","Variance of E_{test}")

test_data= [mvnrnd(mu1,cov1,N); mvnrnd(mu2,cov2,N)];
test_label = [ones(N,1); zeros(N,1)];
N_D = [3, 5, 10, 50, 75, 90];
N_T = N - N_D;
for l =1:loop
   for i = 1:length(N_D)
      ind_train = [randsample(N, N_D(i)); randsample(N, N_D(i))+N];
      ind_test = [randsample(N, N_T(i)); randsample(N, N_T(i))+N];
      train_data = design_data(ind_train,:);
      train_label = design_label(ind_train);
      test_data_subset = test_data(ind_test,:);
      test_label_subset = test_label(ind_test);
      err_design(i,l) = mean(abs(train_label - round(predict(g_classifer, train_data))));
      err_test(i,l) = mean(abs(test_label_subset - round(predict(g_classifer,
test_data_subset))));
   end
end
err_design_mean = mean(err_design,2);
err_test_mean = mean(err_test,2);
err_result = [err_design_mean, err_test_mean]
```

Q3 Codes

```
N_D = [3, 5, 10, 20, 50, 100, 200];
d = [5, 10, 15];
loop = 10;
```

```
N = 500;
err_test_mean = zeros(length(d),length(N_D));
err_test = zeros(length(N_D),loop);
err_true = zeros(length(d),1);
for i = 1:length(d)
   if(i == 1)
      mu1 = (2.15-0.3+0.045)*rand(1,d(i));%[2.5, zeros(1, d(i)-1)];
   elseif(i == 2)
      mu1 = (2.15-0.3+0.05)*rand(1,d(i));
   elseif(i ==3)
      mu1 = (2.15-3*0.3)*rand(1,d(i));
   end
   mu2 = zeros(1,d(i));
   cov = eye(d(i));
   rng(37)  % For reproducibility
   design_data = [mvnrnd(mu1,cov,N); mvnrnd(mu2,cov,N)];
   design_label = [ones(N,1); zeros(N,1)];
   test_data = [mvnrnd(mu1,cov,N); mvnrnd(mu2,cov,N)];
   test_label = [ones(N,1); zeros(N,1)];
   g_classifer = fitcnb(design_data,design_label);
   err_true(i) = mean(abs(test_label - round(predict(g_classifer, test_data))));
   for j = 1:length(N_D)
      for l = 1:loop
         ind = [randsample(N, N_D(j)); randsample(N, N_D(j))+N];
         train_data = design_data(ind,:);
         train_label = design_label(ind);
         g_classifer = fitcnb(train_data,train_label);
         err_test(j,l) = mean(abs(test_label - round(predict(g_classifer, test_data))));
      end
   end
   err_test_mean(i,:) = mean(err_test,2);
end
figure(1)
hold on
for i = 1:length(d)
   plot(N_D,err_test_mean(i,:), 'x-','Linewidth',2,'Markersize',8)
end
xlabel("Size of Training Set")
ylabel("Average Error")
legend("Dimension 5","Dimension 10", "Dimension 15")
```

Q4 Codes

```
mu1 = [3, 2];
mu2 = [6, 5];
cov1 = [4, 2; 2, 4];
cov2 = [4, 2; 2, 4];
d = [0,0;1,1;2,2;3,3;4,4];
N = 500;
```

```
loop = 10;
rng(3)  % For reproducibility
d_mahal = zeros(length(d),1);
err_design = zeros(length(d),loop);
err_test = zeros(length(d),loop);
for i = 1:length(d)
design_data = [mvnrnd(mu1,cov1,N); mvnrnd(mu2+d(i,:),cov2,N)];
design_label = [ones(N,1); zeros(N,1)];
test_data = [mvnrnd(mu1,cov1,N); mvnrnd(mu2+d(i,:),cov2,N)];
test_label = [ones(N,1); zeros(N,1)];
d_mahal(i) = mahal(mu2+d(i,:),design_data(1:N,:));
for j = 1:loop
  g_classifer = fitcnb(design_data,design_label);
  err_design(i,j) = mean(abs(design_label - round(predict(g_classifer, design_data))));
  err_test(i,j) = mean(abs(test_label - round(predict(g_classifer, test_data))));
end
end
err_design_mean = mean(err_design,2);
err_test_mean = mean(err_test,2);
figure(1)
hold on
plot(d_mahal,err_design_mean, 'rx-','Linewidth',2,'Markersize',8)
plot(d_mahal,err_test_mean, 'bx-','Linewidth',2,'Markersize',8)
xlabel("Mahalonobis Distance")
ylabel("Average Error")
legend("Mean of E_{train}","Mean of E_{test}")
```